General Principles of Discrete-Event Simulation Systems

How they work

Radu T. Trîmbiţaş

Babes-Boltai University

1st Semester 2010-2011

Review of Basic Concepts I

- **System** – A collection of entities (e.g., people and machines) that interact together over time to accomplish one or more goals.
- **Model** – An abstract representation of a system, usually containing structural, logical, or mathematical relationships which describe a system in terms of state, entities and their attributes, sets, processes, events, activities, and delays.
- **System state** – A collection of variables that contain all the information necessary to describe the system at any time.
- **Entity** – Any object or component in the system which requires explicit representation in the model (e.g., a server, a customer, a machine).
- **Attributes** – The properties of a given entity (e.g., the priority of a waiting customer, the routing of a job through a job shop).

Review of Basic Concepts II

- **List** – A collection of (permanently or temporarily) associated entities, ordered in some logical fashion (such as all customers currently in a waiting line, ordered by first come, first served, or by priority).
- **Event** – An instantaneous occurrence that changes the state of a system (such as an arrival of a new customer).
- **Event notice** – A record of an event to occur at the current or some future time, along with any associated data necessary to execute the event; at a minimum, the record includes the event type and the event time.
- **Event list** – A list of event notices for future events, ordered by time of occurrence; also known as the future event list (FEL).
- **Activity** – A duration of time of specified length (e.g., a service time or interarrival time), which is known when it begins (although it may be defined in terms of a statistical distribution).

Review of Basic Concepts III

- **Delay** – Aduration of time of unspecified indefinite length, which is not known until it ends (e.g., a customer's delay in a last-in, first-out waiting line which, when it begins, depends on future arrivals).
- **Clock** – A variable representing simulated time, called *CLOCK* in the examples to follow.

Activity I

- An activity typically represents a service time, an interarrival time, or any other processing time whose duration has been characterized and defined by the modeler. An activity's duration may be specified in a number of ways:
  1. Deterministic—for example, always exactly 5 minutes;
  2. Statistical—for example, as a random draw from among 2, 5, 7 with equal probabilities;
  3. A function depending on system variables and/or entity attributes—for example, loading time for an iron ore ship as a function of the ship's allowed cargo weight and the loading rate in tons per hour.

Activity II

- However it is characterized, the duration of an activity is computable from its specification at the instant it begins. Its duration is not affected by the occurrence of other events (unless, as is allowed by some simulation packages, the model contains logic to cancel an event).
- To keep track of activities and their expected completion time, at the simulated instant that an activity duration begins, an event notice is created having an event time equal to the activity's completion time.
- *Example*: if the current simulated time is $CLOCK = 100$ minutes and an inspection time of exactly 5 minutes is just beginning, then an event notice is created that specifies the type of event (an end of inspection event) and the event time ($100 + 5 = 105$ minutes).

Delay I

- In contrast to an activity, a delay's duration is not specified by the modeler ahead of time, but rather is determined by system conditions. Quite often, a delay's duration is measured and is one of the desired outputs of a model run.
- Typically, a delay ends when some set of logical conditions becomes true or one or more other events occur. For example, a customer's delay in a waiting line may be dependent on the number and duration of service of other customers ahead in line as well as the availability of servers and equipment.
- A delay is sometimes called a *conditional wait*, while an activity is called an *unconditional wait*.

Delay II

- The completion of an activity is an event, often called a *primary event*, that is managed by placing an event notice on the *FEL*.
- In contrast, delays are managed by placing the associated entity on another list, perhaps representing a waiting line, until such time as system conditions permit the processing of the entity.
- The completion of a delay is sometimes called a *conditional* or *secondary event*, but such events are not represented by event notices, nor do they appear on the *FEL*.
- The systems considered here are dynamic—that is, changing over time. Therefore, system state, entity attributes and the number of active entities, the contents of sets, and the activities and delays currently in progress are all functions of time and are constantly changing over time. Time itself is represented by a variable called *CLOCK*.

Definition of DES

- The definition of the model components provides a static description of the model.
- In addition, a description of the dynamic relationships and interactions between the components is also needed. Some questions that need answers include:
  1. How does each event affect system state, entity attributes, and set contents?
  2. How are activities defined (i.e., deterministic, probabilistic, or some other mathematical equation)? What event marks the beginning or end of each activity? Can the activity begin regardless of system state, or is its beginning conditioned on the system being in a certain state? (For example, a machining "activity" cannot begin unless the machine is idle, not broken, and not in maintenance.)
  3. Which events trigger the beginning (and end) of each type of delay? Under what conditions does a delay begin, or end?
  4. What is the system state at time 0? What events should be generated at time 0 to "prime" the model—that is, to get the simulation started?

Definition of DES II

- A discrete-event simulation is the modeling over time of a system all of whose state changes occur at discrete points in time—those points when an event occurs.
- A discrete-event simulation (hereafter called a simulation) proceeds by producing a sequence of system snapshots (or system images) which represent the evolution of the system through time.
- A given snapshot at a given time ($CLOCK = t$) includes not only the system state at time t , but also a list (the FEL) of all activities currently in progress and when each such activity will end, the status of all entities and current membership of all sets, plus the current values of cumulative statistics and counters that will be used to calculate summary statistics at the end of the simulation.
- A prototype system snapshot is shown in Figure 1.

Prototype system snapshot

Clock	*System State*	*Entities and Attributes*	*Set 1*	*Set 2*	...	*Future Event List, FEL*	*Cumulative Statistics and Counters*
t	$(x, y, z, \ldots)$					$(3, t_1)$– Type 3 event to occur at time t_1 $(1, t_2)$– Type 1 event to occur at time t_2	

Figure: Prototype system snapshot at simulation time t

Event-scheduling/time-advance algorithm I

- The sequence of actions which a simulator (or simulation language) must perform to advance the clock and build a new system snapshot is called the event-scheduling/time-advance algorithm.
- The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the future event list (*FEL*).
- At any given time t, the *FEL* contains all previously scheduled future events and their associated event times (called t_1, t_2, ...). The *FEL* is ordered by event time, meaning that the events are arranged chronologically; that is, the event times satisfy

$$t < t_1 < t_2 < t_3 < \cdots < t_n$$

- Time t is the value of CLOCK, the current value of simulated time.
- event associated with time t_1 is called the *imminent event*

Event-scheduling/time-advance algorithm II

- After the system snapshot at simulation time $CLOCK = t$ has been updated, the $CLOCK$ is advanced to simulation time $CLOCK = t_1$, and the imminent event notice is removed from the FEL and the event executed.
- Execution of the imminent event means that a new system snapshot for time t_1 is created based on the old snapshot at time t and the nature of the imminent event. At time t_1, new future events may or may not be generated, but if any are, they are scheduled by creating event notices and putting them in their proper position on the FEL.
- After the new system snapshot for time t_1 has been updated, the clock is advanced to the time of the new imminent event and that event is executed.
- This process repeats until the simulation is over.

Event-scheduling/time-advance algorithm III

Step 1. Remove the event notice for the imminent event from *FEL*

Step 2. Advance *CLOCK* to imminent event time (i.e., advance *CLOCK* from t to t_1).

Step 3. Execute imminent event: update system state, change entity attributes, and set membership as needed.

Step 4. Generate future events (if necessary) and place their event notices on *FEL* ranked by event time. (Example: Event 4 to occur at time t, where $t_2 < t < t_3$.)

Step 5. Update cumulative statistics and counters.

Stopping Event

- Every simulation must have a *stopping event*, here called E, which defines how long the simulation will run. There are generally two ways to stop a simulation:
  1. At time 0, schedule a stop simulation event at a specified future time T_E. Thus, before simulating, it is known that the simulation will run over the time interval $[0, TE]$. Example: Simulate a job shop for $T_E = 40$ hours.
  2. Run length T_E is determined by the simulation itself. Generally, T_E is the time of occurrence of some specified event E. Examples: T_E is the time of the 100th service completion at a certain service center. T_E is the time of breakdown of a complex system. T_E is the time of disengagement or total kill (whichever occurs first) in a combat simulation. T_E is the time at which a distribution center ships the last carton in a day's orders.
- In case 2, T_E is not known ahead of time. Indeed, it may be one of the statistics of primary interest to be produced by the simulation.

World views of Model for simulation (Three Types)

- When using a simulation package or even when doing a manual simulation, a modeler adopts a world view or orientation for developing a model:
- *event scheduling* – a simulation analyst concentrates on events and their effect on system state
- *process interaction* – (like processes in OS)
  - We define the model in terms of entities or objects and their life cycle of an entity
  - It has intuitive appeal and allow to describe the process flow in terms of high level block or network constructs
  - Event scheduling is hidden
- These two approaches use a variable time advance; that is, when all events and system state changes have occurred at one instant of simulated time, the simulation clock is advanced to the time of the next imminent event on the FEL.

World views of Model for simulation (Three Types)

Polling and Interrupt

- *activity scanning* – uses a fixed time increment and a rule-based approach to decide whether any activities can begin at each point in simulated time. Slow!
- At each clock advance the conditions for each activity are checked and if they are true then corresponding activity begins
- It is suitable for small system
- Improvement: *three-phase approach*

Three-Phase Approach

- Events are considered to be activities of duration 0 time units
- Classification of activities: **B activities** = activities bound to occur (primary events and unconditional activities) and **C activities** = activities or events that are conditional upon certain condition being true.
- B-activities and events can be scheduled ahead of time $\Longrightarrow$ this allows variable time advance. The FEL contains only B-type activities .
- Scanning to detect whether any C-type activity can begin or C-type events occur happens only at the end of each time advance, after all B-type events have completed.

Three-Phase Approach II

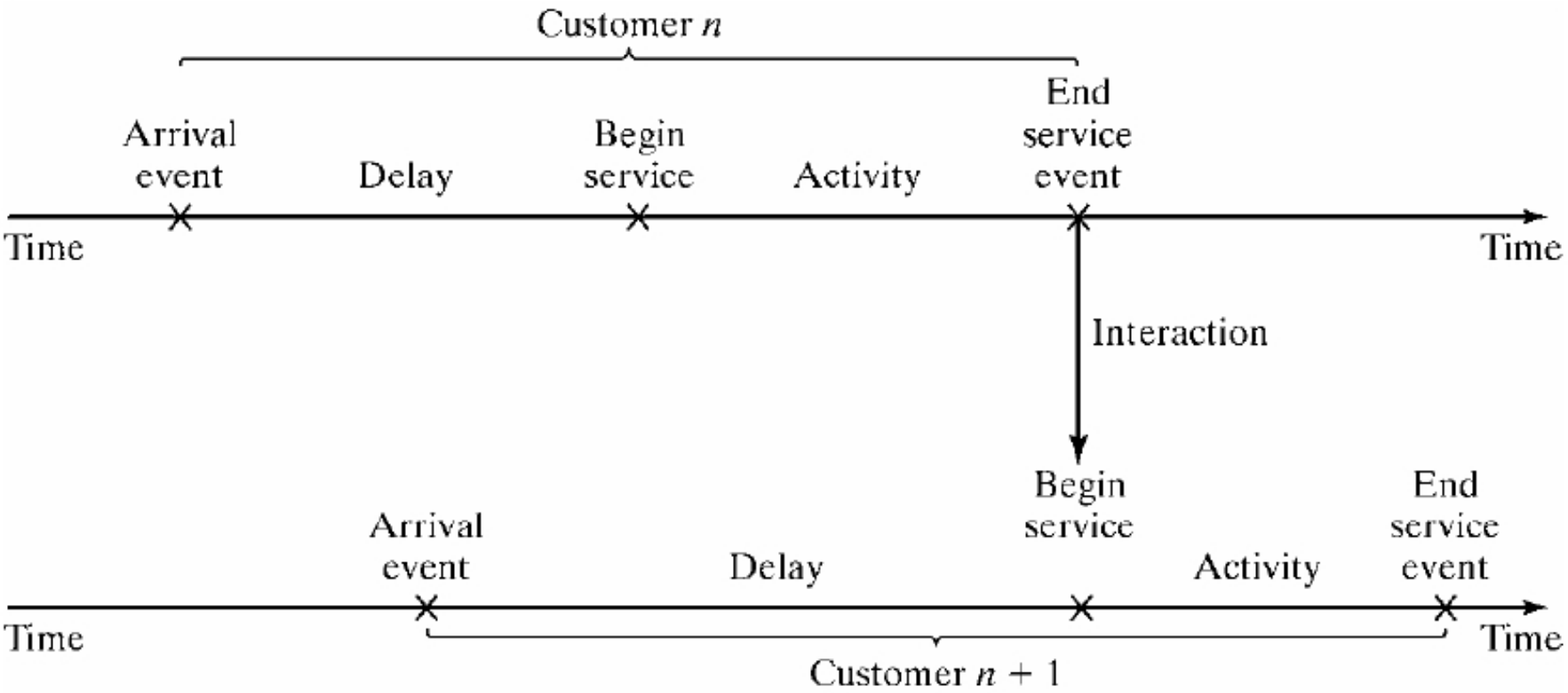
The simulation proceeds with repeated execution of the 3 phases until it is completed:

Phase A Remove the imminent event from the FEL and advance the clock to its event time. Remove from the FEL any other events that have the same event time.

Phase B Execute all B-type events that were removed from the FEL. (this could free a number of resources or otherwise change system state.)

Phase C Scan the conditions that trigger each C-type activity and activate any whos conditions are met. Rescan until no additional C-type activities can begin and no event occur.

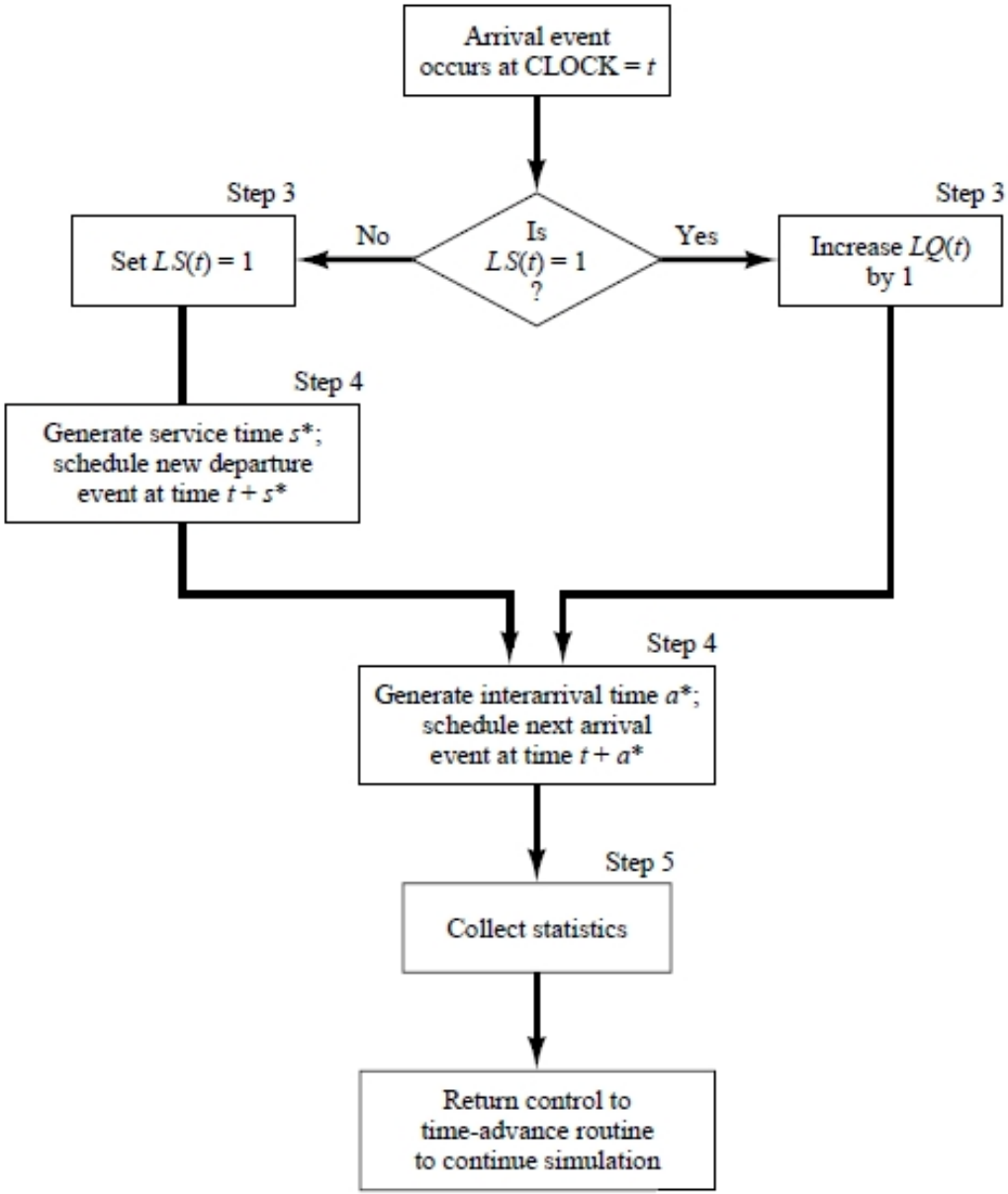
Two customer processes interaction in single server queue



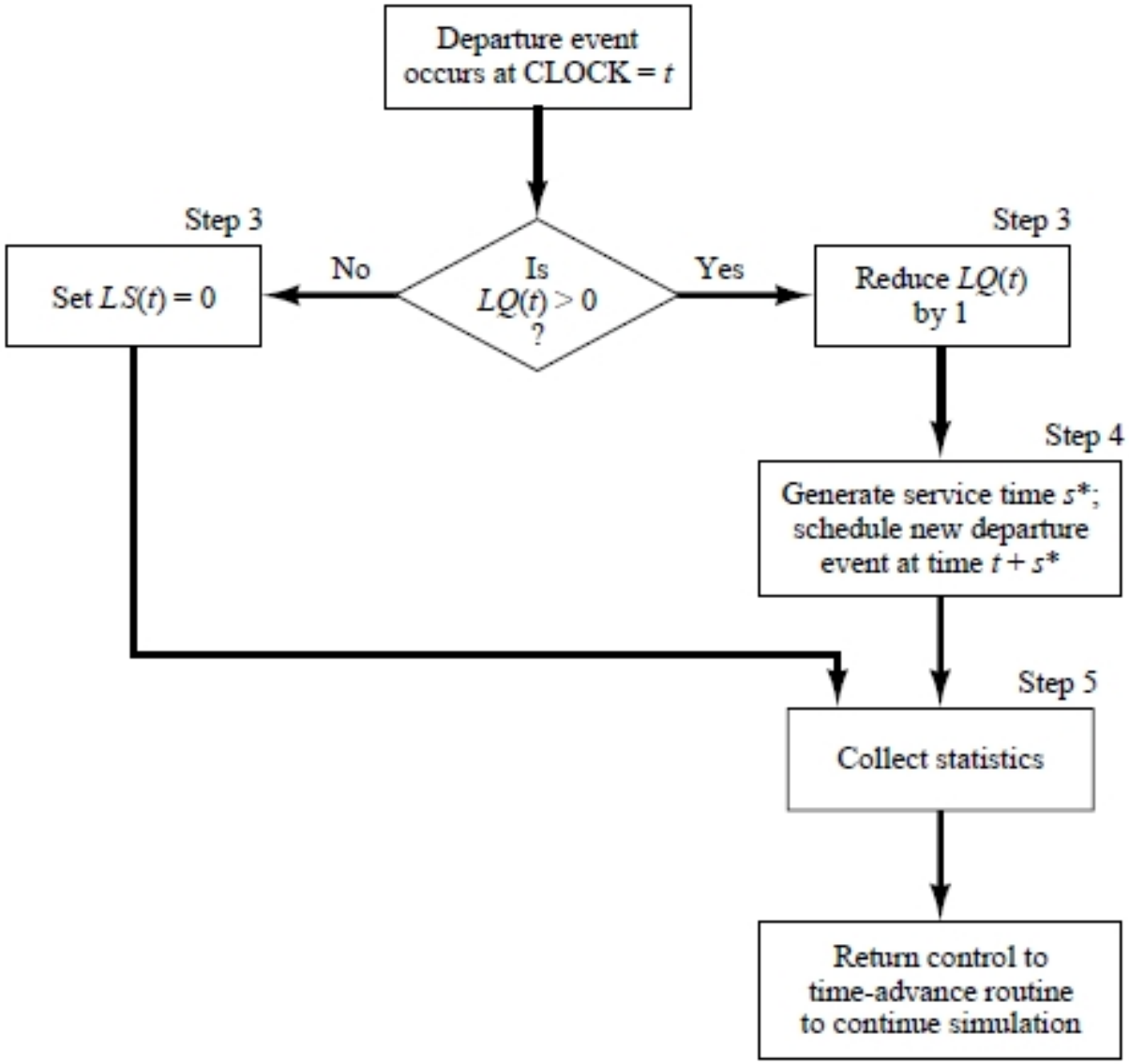
Event Scheduling example (Grocery Center)

- System State
  - $LQ(t)$, $LS(t)$
- Entities
  - The server and customer are not explicitly modeled
- Events
  - Arrival (A), Departure (D), Stopping event ($E = 60$)
- Event notices
  - (A, t), (D, t), $(E, 60)$
- Activities
  - Inter-arrival time, service time
- Delay
  - Customer time spent in waiting time

Execution of the arrival event



Execution of the departure event



Simulation Table

	System State				*Cumulative Statistics*	
Clock	$LQ(t)$	$LS(t)$	*Future Event List*	*Comment*	B	MQ
0	0	1	$(D, 4)$ $(A, 8)$ $(E, 60)$	First A occurs $(a^* = 8)$ Schedule next A $(s^* = 4)$ Schedule first D	0	0
4	0	0	$(A, 8)$ $(E, 60)$	First D occurs: $(D, 4)$	4	0
8	0	1	$(D, 9)$ $(A, 14)$ $(E, 60)$	Second A occurs: $(A, 8)$ $(a^* = 6)$ Schedule next A $(s^* = 1)$ Schedule next D	4	0
9	0	0	$(A, 14)$ $(E, 60)$	Second D occurs: $(D, 9)$	5	0
14	0	1	$(A, 15)$ $(D, 18)$ $(E, 60)$	Third A occurs: $(A, 14)$ $(s^* = 4)$ Schedule next D	5	0
15	1	1	$(D, 18)$ $(A, 23)$ $(E, 60)$	Fourth A occurs: $(A, 15)$ (Customer delayed)	6	1
18	0	1	$(D, 21)$ $(A, 23)$ $(E, 60)$	Third D occurs: $(D, 18)$ $(s^* = 3)$ Schedule next D	9	1
21	0	0	$(A, 23)$ $(E, 60)$	Fourth D occurs: $(D, 21)$	12	1

Computing Mean Response Time (cont.)

- Entities
  - (C_i, t), representing customer C_i who arrive at time t
- Event notices
  - (A, t, C_i), the arrival of customer C_i at future time t
  - (D, t, C_j), the departure of customer C_j at future time t
- Set
  - "*CHECKOUTLINE*" the set of all customers currently at the checkout counter, ordered by time of arrival
- Response time
  - *CLOCKTIME* - attribute "time of arrival"
- S: sum of customer response time for all customers
- N_D: total number of departures up to current time
- F: Total number of customers who spend ≥ 5 minutes in system

Simulation Table

	System State		List	Future Event	Cumulative Statistics		
Clock	$LQ(t)$	$LS(t)$	"CHECKOUT LINE"	List	S	N_D	F
0	0	1	$(C1, 0)$	$(D, 4, C1)$ $(A, 8, C2)$ $(E, 60)$	0	0	0
4	0	0		$(A, 8, C2)$ $(E, 60)$	4	1	1
8	0	1	$(C2, 8)$	$(D, 9, C2)$ $(A, 14, C3)$ $(E, 60)$	4	1	1
9	0	0		$(A, 14, C3)$ $(E, 60)$	5	2	1
14	0	1	$(C3, 14)$	$(A, 15, C4)$ $(D, 18, C3)$ $(E, 60)$	5	2	1
15	1	1	$(C3, 14)$ $(C4, 15)$	$(D, 18, C3)$ $(A, 23, C5)$ $(E, 60)$	5	2	1
18	0	1	$(C4, 15)$	$(D, 21, C4)$ $(A, 23, C5)$ $(E, 60)$	9	3	2
21	0	0		$(A, 23, C5)$ $(E, 60)$	15	4	3

List Processing

- List processing deals with methods for handling lists of entities and the future event list.
- Simulation packages provide, both explicitly for an analyst's use as well as hidden in the simulation mechanism behind the language, facilities for an analyst or the model itself to use lists and perform the basic operations on lists.
- Operations
  1. Removing a record from the top of the list.
  2. Removing a record from any location on the list.
  3. Adding an entity record to the top or bottom of the list.
  4. Adding a record to an arbitrary position on the list, determined by the ranking rule.

Structure of a simulation system